AFRL-AFOSR-VA-TR-2016-0177

Towards a Science Base for Cybersecurity

Fred Schneider
CORNELL UNIVERSITY

06/08/2016
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
03-06-2016	Final Technical	Jun 2011 - Jun 2016

4. TITLE AND SUBTITLE
Towards a Science Base for Cybersecurity

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
FA9550-11-1-0137

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)
Schneider, Fred, B

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Cornell University, 373 Pine Tree Road, Ithaca NY 14850-2820

8. PERFORMING ORGANIZATION REPORT NUMBER
OSP #62480

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research, 875 N Randolph Street, Room 3112, Arlington VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
N/A

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution A - Approved for Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The goal was to better understand approaches for building attack-resistant cyber-systems. This involved implementing new system software, designing formalisms for specifying system security properties, and developing program analysis techniques for enforcing those properties. Two operating systems were built to better understand how trusted coprocessors could be leveraged for increased assurance that unmodified software and applications are executing: Nexus provides support for a desktop and CloudProxy provides support for applications running in cloud. The specification and enforcement of information-use policies that could tag values was also investigated. Here, a theory of RIF (reactive information flow) labels was developed to support re-classification of information as it is transformed by program execution. The theory was then the basis for a new type system, and that type system was retrofit into a programming language.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Fred B. Schneider
U	U	U	UU		19b. TELEPHONE NUMBER *(Include area code)* (607) 255-9221

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

Towards a Science Base for Cybersecurity

AFOSR Grant F9550-11-1-0137

Final Report

15 June 2011 – 14 June 2016

Fred B. Schneider
Computer Science Department
Cornell University
Ithaca, New York
(607) 255-9221 (phone)
fbs@cs.cornell.edu

1 Objectives

The objective was to enable construction of attack-resistant cyber-systems. Various classes of defenses were studied and, in some cases, prototype systems were built. Not only did the prototypes provide insight into classes of defenses, but they provided experience we leveraged for evolving the science base for cyber-security.

Specific topics that we investigated under the auspices of this AFOSR funding included:

- Leveraging trustworthy hardware to increase assurance that unmodified system software and applications are executing. We explored this question both for stand-alone desktop computers and for clouds.
- Understanding use policies that are associated with information, with support for re-classification as a computation proceeds. Information-flow enforcement is based on tagging values with policies that characterize allowed readers or trusted writers; we also explored extensions for other kinds of restrictions on use (including privacy).

2 Summary of Completed Research

2.1 Trusted Computing Implementations

Secure coprocessors, such as industry standard Trusted Platform Modules (TPMs), are becoming ubiquitous. This hardware can be a foundation for software systems that offer strong guarantees about run time behavior. Yet, there is a significant gap between the primitives provided by TPMs and the forms of assurance actually required for applications.

We explored two ways to close that gap:

- We developed a new operating system (Nexus) that employs TPM's. Nexus embodies an authorization architecture that unifies a broad set of approaches for establishing whether a request can be trusted and, thus, should be granted.
- We developed a framework (CloudProxy) that can be deployed—for software at any and all levels of the software stack—the isolation and authentication guarantees that TPMs enable. CloudProxy can be used to protect applications running as tenants in a remote cloud, even if the cloud's operations staff cannot be trusted.

Nexus: Logical Attestation. The key primitive provided by secure coprocessors is *hash-based attestation*, whereby a certificate captures the launch-time hash of components comprising the software stack and associated configuration files. Hash-based attestation forces all trust decisions to be *axiomatic*, because principals are being trusted by fiat. Access control lists that enumerate principals by name, digital signatures to certify that a particular piece of code was vetted by a particular vendor, and authorization based on program hashes are all instances of the axiomatic basis for trust.

An alternative method of establishing trust is to employ an *analysis* that predicts whether certain behaviors by a program are possible. Proof carrying code, in which a program is accompanied by a proof that its execution satisfies certain properties, instantiates this analytical basis for trust. Similarly, systems that employ typecheckers and domain-specific languages, in which code snippets are loaded and executed only if the code is deemed safe, are employing analysis for establishing trust.

Finally, a *synthetic* basis for trust is involved when a program is transformed prior to execution and the transformed artifact, by construction, can be trusted in ways that the original could not. Sandboxing SFI, inlined

reference monitors, and other program rewriting techniques create such a synthetic basis for trust.

Today's operating systems provide disparate mechanisms to implement these three bases of trust. The challenge was to unify them into a single authorization architecture. Nexus does that unification with its *logical attestation* approach to authorization. In logical attestation, a *labeling function* generates an attributed statement called a *label* and expressed in NAL (Nexus Authorization Logic), a constructive logic of beliefs. Labels are unforgeable, machine-parseable statements of the form "P **says** $\mathcal{S}$" that capture information relevant to authorization decisions involving principal P. A bitstring that encodes a label is known as a *credential*. Since labeling functions can be provided by third parties and labels are logical statements, a rich set of properties become available for authorizing access requests. These properties can incorporate references to dynamic system state, including the current time, current resource availability, and even history. Labels used in proofs assert reasons why a principal might be trusted; the proofs are checked by *guards* and constitute the basis for deciding whether to grant or deny a request.

Nexus executes on x86 platforms equipped with a TPM, supports much of the Posix API, and natively executes many Linux applications. It seems to have been the first operating system to implement logic-based authorization with dynamic system state, the first to implement operating system capabilities based on statements issued by a TPM, and the first to support all three bases for trust in a single unified framework.

Logical attestation enables novel authorization functionality and provides strong and useful guarantees today's systems cannot provide. We illustrated its power by developing a cloud computing application, called Fauxbook, that implements guarantees about safety, confidentiality, and resource control. Fauxbook provides a familiar social networking experience, where users publicly post and exchange status messages. The Nexus authorization architecture even blocks Fauxbook developers from examining or data-mining information Fauxbook handles. Moreover, logical attestation enabled the cloud-infrastructure operator to guarantee certain forms of resource availability to Fauxbook developers. Experiments showed that the cost of authentication with logical attestation in Fauxbook is on the order of 1ms, and it can be reduced to 20 cycles with proof caching, an optimization we describe later.

CloudProxy. CloudProxy is a new framework that supports secure deployment of applications to clouds, defends against insider attacks, and provides protocols for automatic key management. Data managed by CloudProxy is never stored or transmitted in unencrypted form, and cryptographic keys are provisioned in a way that defends against malicious operators or other data-center insiders. Protocols are provided for remote or local clients to authenticate the executable and execution environment of a server and for a server to authenticate the executable and execution environment of its clients. Three prototype applications have been implemented to evaluate the utility of CloudProxy: FileProxy, a file service; AuthProxy, an authentication service for remote third parties; and BidProxy, an auction service. Performance measurements demonstrated that CloudProxy is a practical way to support secure, distributed applications.

CloudProxy combines hardware-based memory isolation along with unforgeable *measurement-based security principals* as supported by Trusted Platform Modules (TPMs) and other secure co-processors. Measurement-based security principals associate a cryptographic key with some measurement value. Only principals having that associated measurement value are permitted to access and use the key. The measurement value typically combines the hash of a requesting program's executable with any environment information that affects program execution—for example, boot parameters and information identifying the host execution environment. Consequently, the capability to generate digital signatures or to decrypt data is available only to unmodified programs being executed in unmodified environments.

Specialized hardware is just one way to implement measurement-based security principals, but embodiments of CloudProxy are not limited to the lowest level of a system's software stack. Moreover, CloudProxy can be deployed recursively, because a host system supporting it necessarily has means to enable its hosted programs to instantiate a CloudProxy for programs that they host. For example, we implemented a stack of three levels, each instantiating a CloudProxy: the *Trusted Hardware* (TrHW) supports a CloudProxy for an operating system called *Trusted OS* (TrOS); and TrOS provides a CloudProxy for programs running as *activity elements* which, together, comprise an *activity*. An activity is an instance of a distributed computation executing on behalf of some *activity owner*. And an *activity owner policy* specifies authenticated claims that must accompany a request in order for that request to be deemed authorized.

Cryptography is a key ingredient for CloudProxy. It protects confidentiality and integrity of data stored on secondary storage or sent over to clients of a hosted application. It is used to authenticate activity elements to their

clients. And it is used in *claims-based authorization* for controlling access to activity functionality. So, the CloudProxy framework includes means to provision cryptographic keys, providing a small, independently-deployed component (hence, easily trusted) along with protocols that defend against malicious actions by data-center operators or employees.

2.2 Use Policies

Reactive Information Flow Policies. An *information flow label* is a tag that gives restrictions on the use of a tagged value v and all values derived from v.

- For confidentiality, it specifies which principals can read the tagged value or can read values derived from the tagged value.
- For integrity, it specifies which principals must be trusted in order to trust the tagged value and any values derived from that tagged value.

Notice that information flow labels offer end-to-end guarantees—they specify current and future use of information, regardless of what variable stores that information or how that information was derived. In contrast, access control policies restrict access to specific information containers, independent of what the container stores or how the value it stores was derived.

Restrictions imposed on a derived value v ought to depend on (i) the information flow labels that tag initial values and (ii) the operations involved in deriving v from those initial values. It is naive, however, simply to tag a derived value v with the set of the information flow labels associated with the values from which v was derived and, thus, impose the conjunction of those restrictions. Operations transform their arguments to produce new values, and a given transformation might warrant a *reclassification* because restrictions associated with inputs to the operation no longer apply to the result produced. With a strong cryptosystem, for example, any principal ought to be allowed to read the value of $\mathit{Encrypt}(x, \mathit{key})$ even though only a few principals are allowed to read the values of x and key. So we would be justified in associating weaker restrictions with the output of *Encrypt* operations than were associated with the inputs.

Reactive information flow labels (RIF labels) specify (i) restrictions on the use of a value as well as (ii) how those restrictions change in response to operations that transform the value. Thus, RIF labels make explicit the connection between information transformations and changes to restrictions. For example, a RIF label might assert that only some principal A (say) is

allowed to read values x and key, any principal may read the output of $Encrypt(x, key)$, and only principals that can read key are allowed to read the output of $Decrypt(y, key)$. So $Encrypt(x, key)$ has weaker restrictions than x but $Decrypt(y, key)$ has stronger restrictions than y.

Under the auspices of this AFOSR grant, we derived a theory for RIF labels. We also defined a security condition that makes sense as the goal when values have been tagged with RIF labels. Classical non-interference does not work, since it cannot handle reclassifications that weaken restrictions. *Piecewise noninterference* (PWNI) extends classical noninterference in a way that does allow values to be reclassified in arbitrary ways. We also investigated static enforcement (i.e., compile-time analysis) for programs where variable declarations include RIF labels. Here, we designed a type system whose type correctness implies PWNI.

The type system for RIF labels makes minimal assumptions about the underlying mathematical structures used for defining labels:

- label $L \sqcup L'$ that embodies the restrictions of labels L and L' has a representation as a RIF label and is computable from labels L and L'
- it is decidable whether one label L is more restrictive than some other label L'

Two families of mathematical structures, which satisfy these conditions, have been explored as the basis of RIF labels that seem useful in practice: finite state automata (where states correspond to restrictions and state transitions correspond to operations) and stacks specialized for cryptographic operations (where push and pop are used to record the nesting of the operations and keys used in generating values). The two families can be combined to handle applications that use fully homomorphic encryption.

To demonstrate the practicality and utility of automata-based RIF labels, JRIF, a new dialect of Java was developed. JRIF derives from Myer's Jif compiler and runtime. Jif's labels, which are based on JIF's Decentralized Label Model, were replaced by RIF automata, and Jif's restrictiveness relation on labels was modified accordingly. Our experience in building and using JRIF gives confidence that other languages for information flow control could be extended similarly. We also programmed two JRIF applications that leverage the expressive power of RIF automata: a Battleship game and a shared calendar application. This exercise demonstrated that RIF automata are easy to use. A public release of the source code for the JRIF compiler and runtime, along with the example applications, are available for download from the JRIF web page.

Use-Based Privacy. In response to all the criticisms about notice and consent, there has been a resurgent focus on viewing privacy in terms of limitations on data use. In some cases, the emphasis is placed on preventing harmful uses without explicit user control, in others, the emphasis is on enabling user control over data uses. Since different users are likely to have different opinions regarding what constitutes a privacy violation and since there is no consensus on how to define "harmful" we decided to explore options for enabling user control.

Data use can occur at any point after data is collected, so control over data use naturally aligns with the idea of *policy tags*. Policy tags are labels that travel with a value and express limitations on how that value may be used. Goals that motivated the design of our scheme are:

- **Expressiveness**: Users should be able to control how their data are used as well as what data become known (both by a service provider with which they interact and by third parties).
- **Scalability**: The burden placed on users should be reasonable, even if users interact with many service providers.
- **Transparency**: Privacy policies should be easily understood and transparent. They should clearly specify how observed data and derived values are used.
- **User Policy Revision**: Users should be allowed to revise privacy policies and, thereafter, should enforce the revision.
- **Enforcement**: Some enforcement mechanism ensures policy compliances.

In order to realize these goals, we developed *avenance tags*. Avenance tags use a new language for expressing privacy policies and are handled in the context of an avenance ecosystem. The connection between avanance tags and RIF labels should be clear; and it allows us to leverage insights we have developed for RIF labels.

3 Impacts on the Community

It can be difficult for new ideas to have an immediate impact. However, NSA's funding for its Science of Security Lablets at univerities and, more recently, their initiative—supported by a series of workshops—to define a "science" of privacy have been heavily influenced by our advocacy for these

foundational approaches to security and privacy. (And NSA consulted with the PI extensively about both initiatives.)

There are other less-direct transitions from the PI's involvement in various advisory capacities during the period of this funding.

- Schneider served as Chief Scientist of the NSF TRUST Science and Technology Center, which included U.C. Berkeley, Carnegie-Mellon University, Cornell University, Stanford University, and Vanderbilt University.

- Schneider was a member of the following industrial advisory boards: Accuvant; Fortify Software Technical Advisory Board; Intel Science and Technology Center for Secure Computing; Microsoft's Trustworthy Computing Academic Advisory Board (co-chair); Riskive Technical Advisory Board (chair); and ZeroFox Technical Advisory Board (chair).

- Schneider served on the following other advisory committees: Computer Science and Telecommunications Board, National Academies; Computing Research Association Board of Directors; Computing Community Consortium Council; Cyber Security Research Alliance; Defense Science Board; EPIC Advisory Board, Lincoln Laboratories; Forum on Cyber-Resiliences, National Academies (chair and founder); NSA best scientific cybersecurity paper award panel; Naval Studies Board, National Academies; and NIST Information Security and Privacy Advisory Board.

- Schneider served on the study committee for the following DoD-related reports:

  - Review of U.S. Navy Cyber Defense Capabilities. Naval Studies Board, National Academies.
  - Study on Supply Chain Security. Defense Science Board. In progress.

4 Publications Supported

1. Nexus Authorization Logic. *ACM Transactions on Information and System Security* 14, 1 (2011), Article 8. And Kevin Walsh, Emin Gun Sirer.

2. NetQuery: A Knowledge Plane for Reasoning about Network Properties. *Proceedings of ACM SIGCOMM 2011* (Toronto, Ontario, Canada August 2011), 278–289. With Alan Shieh and Emin Gun Sirer.

3. Logical Attestation: An Authorization Architecture for Trustworthy Computing. *SOSP'11 Proceedings of 23rd ACM Symposium on Operating Systems Principles* (Cascais, Portugal, October 2011), 249–264. With Emin Gun Sirer, Willem De Bruijin, Patrick Reynolds, Alan Shieh, Kevin Walsh, and Dan Williams.

4. A Doctrinal Thesis. Editorial. *IEEE Security & Privacy* 9, 4 (July/August 2011), 3–4. With Deirdre Mulligan.

5. Doctrine for Cybersecurity. *Daedalus*. Fall 2011, 70–92. With Deirdre Mulligan

6. Beyond Traces And Independence. *Dependable and Historic Computing. Essays Dedicated to Brian Randell on the Occasion of His 75th Birthday*, Lecture Notes in Computer Science, Vol. 6875 (Cliff Jones and John Lloyd, eds). Springer Verlag, 2011, 479–485.

7. Computing researchers get 'schooled' on science policy at CCC workshop. *Computing Research News* Volume 24, No. 1 (January 2012). With Peter Harsha.

8. Blueprint for a Science Of Cybersecurity. *The Next Wave* Volume 19, No. 2 (March 2012), 47–57.

9. Breaking-in Research. Editorial. *IEEE Security and Privacy* March/April 2013.

10. Cybersecurity Education in Universities. Editorial. *IEEE Security and Privacy* July/August 2013.

11. Federated Identity Management Systems: A Privacy-based Characterization. *IEEE Security and Privacy* 11, 5 September/October 2013, 36–48.

12. The CloudProxy Tao for Trusted Computing. Preliminary version available as University of California, Berkeley Technical Report No, UCB/EECS-2013-135, July 2013. With John Manferdelli and Tom Roeder.

13. When Not All Bits Are Equal: Incorporating "Worth" into Information-Flow Measures. *POST 2014 Principles of Security and Trust* (Grenoble, France, April 2014) Lecture Notes in Computer Science, vol 8414. M. Abadi and S. Kremer Eds. 120–139. With Mario Alvim and Andre Scedrov.

14. Incentivizing Quality and Impact: Evaluating Scholarship in Hiring, Tenure, and Promotion. Best Practices Memo, Computing Research Association, Adopted February 2015. With Batya Friedman. `http://www.cra.org/uploads/documents/resources/bpmemos/BP_Memo`

15. Enforcing Privacy Policies with Meta-Code. *6th ACM SIGOPS Asia-Pacific Workshop on Systems*, (Tokyo, Japan, July 2015). With Havard Johansen, Eleanor Birrell, Robbert van Renesse, Magnus Stenhaug, and Dag Johansen.

16. Vive La Difference: Paxos vs. Viewstamped Replication vs. Zab. *IEEE Transactions on Dependable and Secure Computing* 12, 4 (July-Aug. 2015), 472–484. With Robbert van Renesse and Nicolas Schiper.

17. Omni-Kernel: An Operating System Architecture for Pervasive Monitoring and Scheduling. *IEEE Transactions on Parallel & Distributed Systems* 26, 10 (October 2015), 2849–2862. With Age Kvalnes, Dag Johansen, Robbert van Renesse, and Steffen Valvag.

18. JRIF: Reactive Information Flow Control for Java. Submitted for publication. Preliminary version available as eCommons technical report 1813/41194, Oct 24, 2015. With Elisavet Kozyri, Owen Arden, Andrew C. Myers.

1.

1. Report Type

Final Report

Primary Contact E-mail

Contact email if there is a problem with the report.

fbs@cs.cornell.edu

Primary Contact Phone Number

Contact phone number if there is a problem with the report

607-255-9221

Organization / Institution name

Cornell University

Grant/Contract Title

The full title of the funded effort.

Towards a Science Base for Cybersecurity

Grant/Contract Number

AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-11-1-0137

Principal Investigator Name

The full name of the principal investigator on the grant or contract.

Fred B. Schneider

Program Manager

The AFOSR Program Manager currently assigned to the award

Tristan Nguyen

Reporting Period Start Date

06/01/2011

Reporting Period End Date

06/14/2015

Abstract

The goal was to better understand approaches for building attack-resistant cyber-systems. This involved implementing new system software, designing formalisms for specifying system security properties, and developing program analysis techniques for enforcing those properties. Two operating systems were built to better understand how trusted coprocessors could be leveraged for increased assurance that unmodified software and applications are executing: Nexus provides support for a desktop and CloudProxy provides support for applications running in cloud. The specification and enforcement of information-use policies that could tag values was also investigated. Here, a theory of RIF (reactive information flow) labels was developed to support re-classification of information as it is transformed by program execution. The theory was then the basis for a new type system, and that type system was retrofit into a programming language.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

DISTRIBUTION A: Distribution approved for public release.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF The maximum file size for an SF298 is 50MB.

005.SF298.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

003.final.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

Nexus Authorization Logic.
ACM Transactions on Information and System Security 14, 1 (2011), Article 8.
And Kevin Walsh, Emin Gun Sirer.

NetQuery: A Knowledge Plane for Reasoning about Network Properties.
Proceedings of ACM SIGCOMM 2011 (Toronto, Ontario, Canada August 2011), 278--289.
With Alan Shieh and Emin Gun Sirer.

Logical Attestation: An Authorization Architecture for Trustworthy Computing.
SOSP'11 Proceedings of 23rd ACM Symposium on Operating Systems Principles (Cascais, Portugal, October 2011), 249--264.
With Emin Gun Sirer, Willem De Bruijin, Patrick Reynolds, Alan Shieh, Kevin Walsh, and Dan Williams.

A Doctrinal Thesis.
Editorial. IEEE Security \& Privacy 9, 4 (July/August 2011), 3--4.
With Deirdre Mulligan.

Doctrine for Cybersecurity.
Daedalus. Fall 2011, 70--92.
With Deirdre Mulligan

Beyond Traces And Independence.
Dependable and Historic Computing. Essays Dedicated to Brian Randell on the Occasion of His 75th Birthday}, Lecture Notes in Computer Science, Vol. 6875 (Cliff Jones and John Lloyd, eds). Springer Verlag, 2011, 479--485.

Computing researchers get 'schooled' on science policy at CCC workshop.
Computing Research News Volume 24, No. 1 (January 2012).
With Peter Harsha.

Blueprint for a Science Of Cybersecurity.
The Next Wave Volume 19, No. 2 (March 2012), 47--57.

Breaking-in Research.
Editorial. IEEE Security and Privacy, March/April 2013.

Cybersecurity Education in Universities.
Editorial.IEEE Security and Privacy, July/August 2013.

Federated Identity Management Systems:
A Privacy-based Characterization.
IEEE Security and Privacy 11, 5 September/October 2013, 36--48.

The CloudProxy Tao for Trusted Computing.
Preliminary version available as University of California, Berkeley Technical Report No, UCB/EECS-2013-135, July 2013.
With John Manferdelli and Tom Roeder.

When Not All Bits Are Equal: Incorporating "Worth" into Information-Flow Measures.
POST 2014 Principles of Security and Trust (Grenoble, France, April 2014)
Lecture Notes in Computer Science, vol 8414. 120--139.
With Mario Alvim and Andre Scedrov.

Incentivizing Quality and Impact: Evaluating Scholarship in Hiring, Tenure, and Promotion.
Best Practices Memo, Computing Research Association, Adopted February 2015.
\url{http://www.cra.org/uploads/documents/resources/bpmemos/BP_Memo}
With Batya Friedman.

Enforcing Privacy Policies with Meta-Code.
6th ACM SIGOPS Asia-Pacific Workshop on Systems (Tokyo, Japan, July 2015).
With Havard Johansen, Eleanor Birrell, Robbert van Renesse, Magnus Stenhaug, and Dag Johansen.

Vive La Difference: Paxos vs.~Viewstamped Replication vs.~Zab.
IEEE Transactions on Dependable and Secure Computing} 12, 4 (July-Aug. 2015), 472--484.
With Robbert van Renesse and Nicolas Schiper.

Omni-Kernel: An Operating System Architecture for Pervasive Monitoring and Scheduling.
IEEE Transactions on Parallel \& Distributed Systems 26, 10 (October 2015), 2849--2862.
With Age Kvalnes, Dag Johansen, Robbert van Renesse, and Steffen Valvag.

JRIF: Reactive Information Flow Control for Java.
Submitted for publication.
Preliminary version available as eCommons technical report 1813/41194, Oct 24, 2015.
With Elisavet Kozyri, Owen Arden, Andrew C. Myers.

2. New discoveries, inventions, or patent disclosures:

Do you have any discoveries, inventions, or patent disclosures to report for this period?

No

Please describe and include any notable dates

Do you plan to pursue a claim for personal or organizational intellectual property?

Changes in research objectives (if any):

None.

Change in AFOSR Program Manager, if any:

None.

Extensions granted or milestones slipped, if any:

None.

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Jun 08, 2016 10:19:31 Success: Email Sent to: fbs@cs.cornell.edu